

# Perceptron

 [medium.com/@manveetdn/notes-on-perceptron-padhai-onefourthlabs-course-a-first-course-on-deep-learning-ed0cad95380e](https://medium.com/@manveetdn/notes-on-perceptron-padhai-onefourthlabs-course-a-first-course-on-deep-learning-ed0cad95380e)

**Disclaimer:** This is notes on “Perceptron” Lesson (PadhAI onefourthlabs course “A First Course on Deep Learning”)



## Disadvantage of the MP-Neuron:

$$y = m \cdot x + b$$

1. **Boolean input** and **Boolean outputs** only
2. **Linear**(only Linear equation sin. cosine, functions are not possible)
3. **Fixed Slope**(Slope is fixed that might be -1 or so)
4. **Few Intercepts** (only one b (bias))

The main reason behind perceptron model to suffice the need of giving non-boolean values as the inputs a better equation to classify more intercepts and a variable slope.

## Example : Oil Drilling Case(mining)

It depends on different factors density, salinity, pressure, depth of the oceans surface these all are non boolean values and we want to taken real values and we want to take real values instead of converting values to boolean.

## Perceptron:

Perceptron satisfies some of the restrictions faced by the us in MP neurons the main difference are as below mentioned.

	MP Neuron	Perceptron
Data	Boolean inputs	Real inputs
Classification	Boolean outputs	Boolean outputs
Model	Linear, only one parameter $b$	Weights for every input, Linear
Loss	$= \sum_i (y_i - \hat{y}_i)^2$	$\sum_i \max(0, 1 - y_i \times \hat{y}_i)$
Learning	Brute force	Principles try to update the weight
Evaluation	Accuracy = $\frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$	same as MP Neuron

Differences between MP-Neuron and Perceptron.

Perceptron:

---

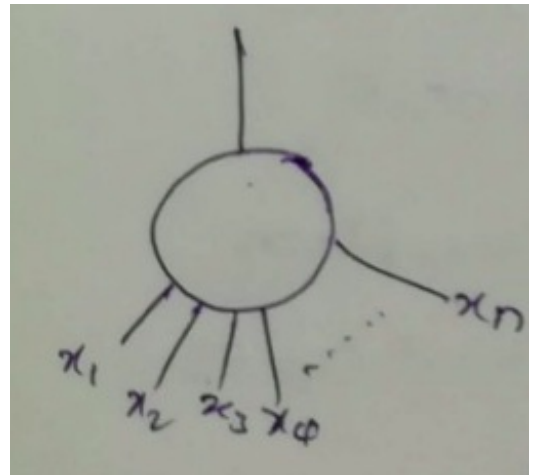
Data and Task:

---

Perceptron model can even have boolean values but also have real values mainly in the case of this perceptron model.

## Perceptron model

In the perceptron model taking real data we will take many real data we will take many real values too high values too low values in this computations is complex so actually we will standardise the data by a formula as below.



	$x' = \frac{x - \min}{\max - \min}$	
Prev	after standardization	
Ex:-		
5.8	0.64	
6.18	0.82	
5.84	0.67	
6.2	0.88	
5.9	0.7	
4.7	0.47	
6.41	1	
5.5	0.47	
		$\min = 4.7$
		$\max = 6.41$
		$\therefore x'$ is calculated and data is normalise as beside before and after

Standardisation of data.

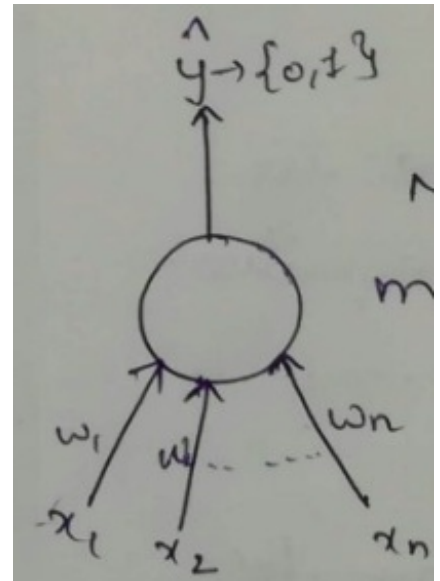
Like this we will standardise the data and we will make data more efficient.

But in perceptron model finally even if the inputs are real the outputs are real the outputs will be boolean.

The Model:

## Perceptron.

Its more or less equal to the MP neuron model in diagram the main difference is here all  $x_1, x_2, x_3, x_4, \dots, x_n$  are real values and also  $w_1, w_2, w_3, \dots, w_n$  are the weight assigned to each value.



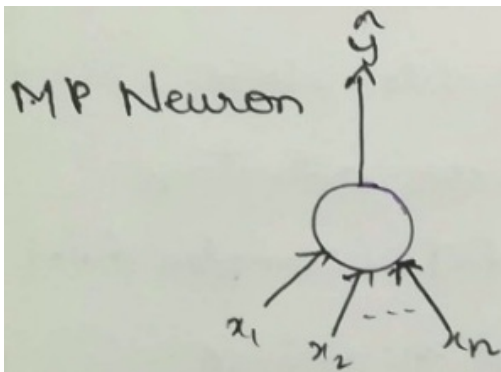
$$\hat{y} = 1 \text{ if } \sum_{i=1}^n w_i x_i \geq b$$
$$\hat{y} = 0 \text{ otherwise}$$

## Output Condition.

Like that is the summation is greater than  $b$  then the output is one else the output is zero.

## Difference between MP-Neuron and Perceptron:

---



$$\hat{y} = 1 \quad \sum_{i=1}^n x_i \geq b$$

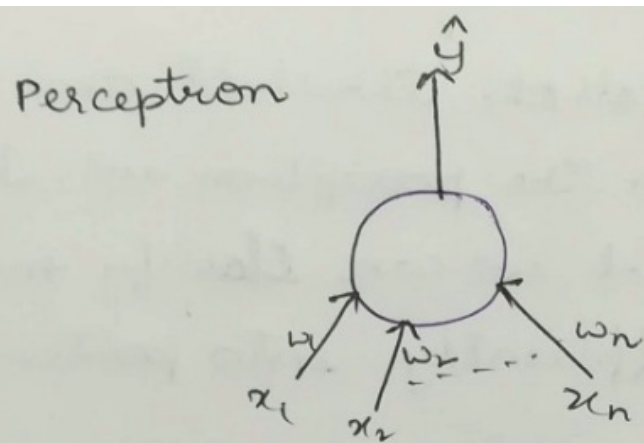
$$\hat{y} = 0 \quad \text{otherwise}$$

Boolean Inputs

Linear

Inputs are not weighted

Adjustable threshold



$$\hat{y} = 1 \quad \text{if} \quad \sum_{i=1}^n w_i x_i \geq b$$

$$\hat{y} = 0 \quad \text{otherwise}$$

Real inputs

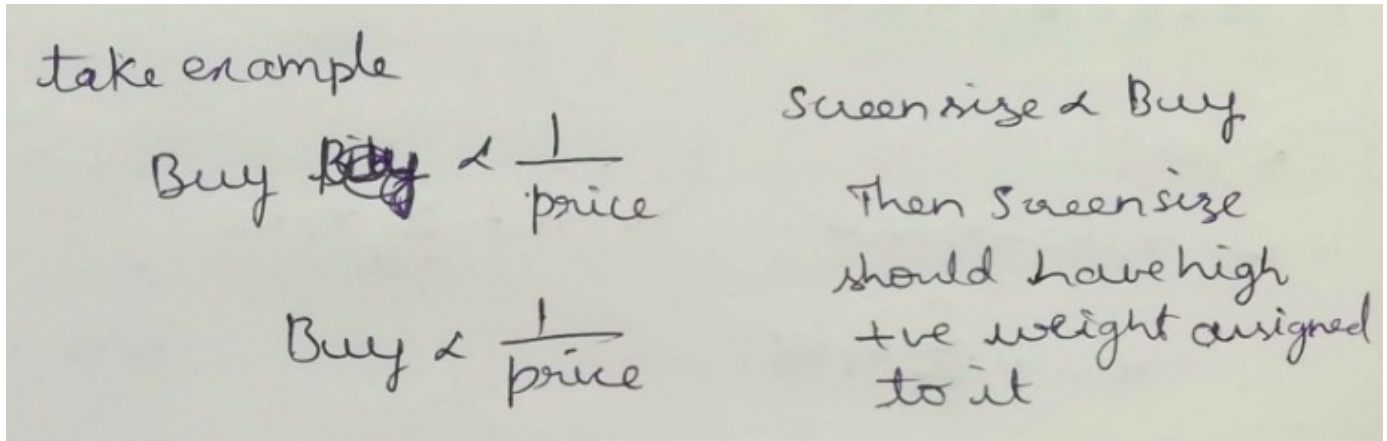
Boolean output

Weights for each input

Adjustable threshold

Difference between MP-Neuron and Perceptron.

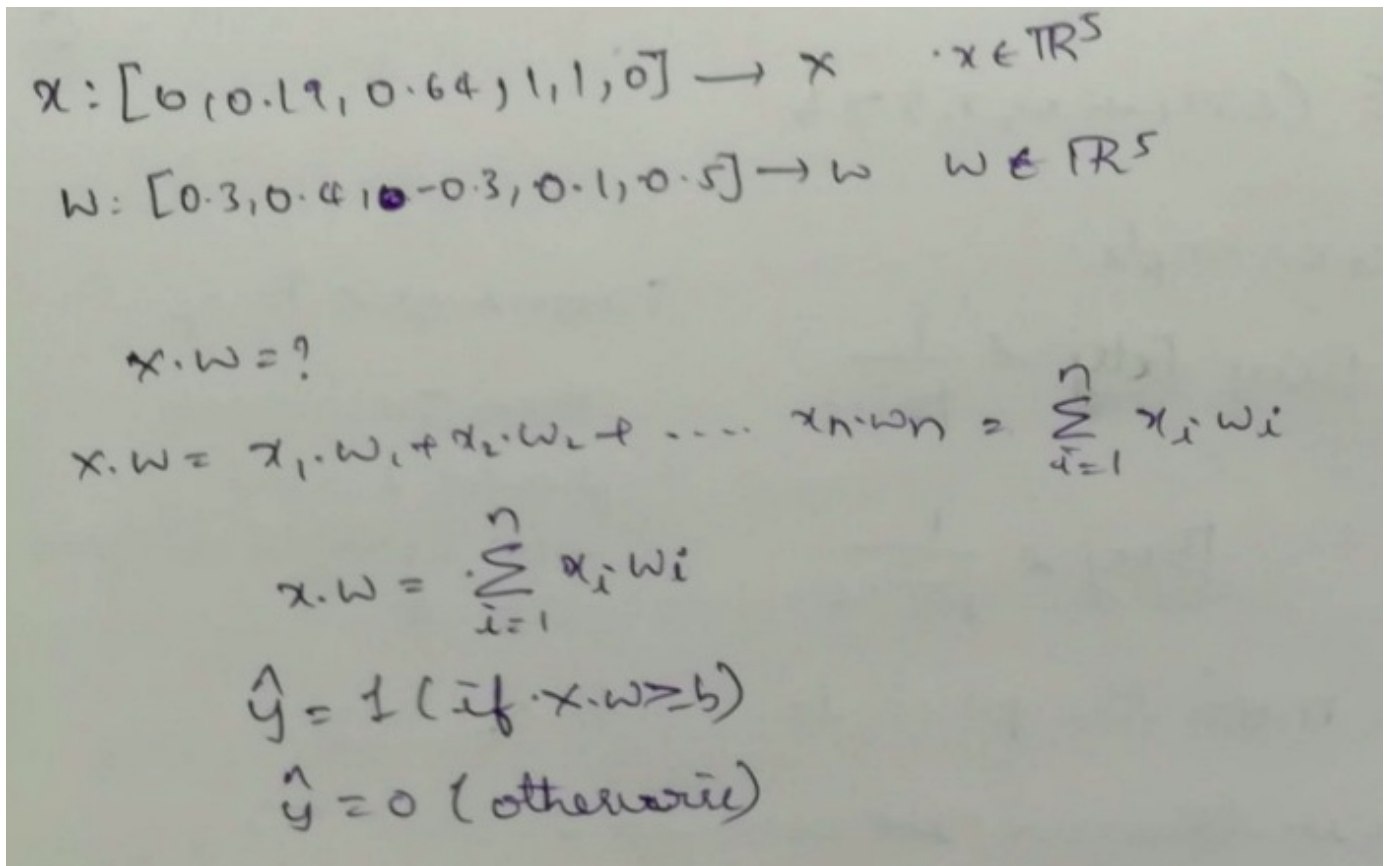
Like that another way of differentiating MP Neuron and the perceptron is



More the price less number of buyers

So in this case we need to reduce the weight and assign negative value for the price then we will update like that.

That's the importance of having weights in a model they allow you to decide the importance of each feature and also have a negative importance for each features.



Predict the output of the process

Geometric Interpretation of the model:



For MP neuron

$$\hat{y} = (x_1 + x_2 \geq b)$$

$$x_1 + x_2 - b \geq 0$$

$$x_2 = -x_1 + b$$

compare

$$y = mx + c$$

$m = -1$

 $c = b$ 

slope is constant

For Perceptron

$$w_1 x_1 + w_2 x_2 = b$$

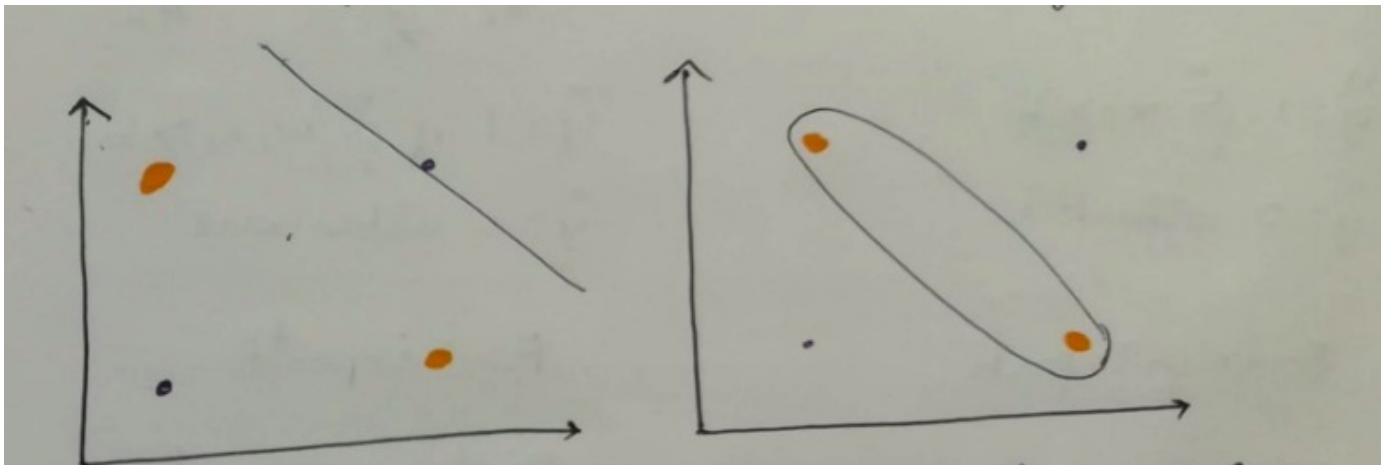
$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{b}{w_2}$$

$$m = -\frac{w_1}{w_2} \quad c = \frac{b}{w_2}$$

slope is not constant

This is the geometric interpretation of the of the MP-Neuron and the perceptron.

With the threshold and the variables slope in the perceptron we have an advantage that we can classify more data (complex data) efficiently into positive and negative.



(i) How MP-Neuron classifies, (ii) How perceptron even cannot classify this model

As shown above it efficiently classifies data compared to MP Neuron but it still lags in fully classifying data.

**For 3-D shape:**

$$w_1 * x_1 + w_2 * x_2 + w_3 * x_3 - b = 0$$

Loss Function:

Here for example purpose if actual output(predicted one) = true output then we will assign a loss else we will assign a loss =1

### Perceptron Loss Function

Purpose of the Loss function is to say the model that some correction need to be done.

That is the perceptron loss function.

Perceptron loss = Square Error Loss

They both are same When we have boolean outputs.

$$L = 0 \text{ if } y = \hat{y} \\ 1, \text{ otherwise}$$

Weight	ScreenSize	Like(y)	$\hat{y}$	Perceptron loss	Square Error loss
0.19	0.64	1	1	0	0
0.63	0.87	1	0	1	1
0.33	0.67	0	1	1	1
1	0.89	0	0	0	0

Learning algorithm:

**General Recipe :P :**



Initialize  $w_1, w_2, b$

Iterate over data

$L = \text{compute\_loss}(\text{~~w_1, w_2, b~~, } x_i)$  Calculate loss

$\text{update}(w_1, w_2, b, L)$  Based on loss update the weights

till satisfied

Iterate and run this above steps till Satisfied that is  $\text{loss} = 0$

Basic Steps

**Algorithm:**

$$\hat{y} = 1 \left( \sum_{i=1}^n w_i x_i \geq b \right)$$

$$w_2 x_2 + w_1 x_1 - b \geq 0$$

$$\text{let } w_0 = -b \cdot x_0 = 1$$

$$w_2 x_2 + w_1 x_1 + \underbrace{w_0 x_0}_{(-b)(1)} \geq 0$$

$$\sum_{i=0}^n w_i x_i \geq 0$$

$$w \cdot x \geq 0$$

$$\therefore \hat{y} = \begin{cases} 1 & (\text{if } w \cdot x \geq 0) \\ 0 & (\text{otherwise}) \end{cases} \quad \left. \vphantom{\begin{matrix} 1 \\ 0 \end{matrix}} \right\} \text{Another way of writing}$$

$$\hat{y} = 1 \left( \text{if } \sum_{i=0}^n w_i \cdot x_i \geq 0 \right)$$

$$\hat{y} = 0 \text{ (otherwise)}$$

Algorithm what we do actually.

## Algorithm : Perceptron learning Algorithm

$P \leftarrow$  input with label 1;

$N \leftarrow$  input with label 0;

Initialise  $w$  randomly;

$P$  is positive side

$N$  is negative side

while !convergence do

! Pick random  $x \in P \cup N$

! if  $x \in P$  and  $\sum_{i=0}^n w_i * x_i < 0$  then

!  $w = w + x$ ;

! end

! if  $x \in N$  and  $\sum_{i=0}^n w_i * x_i \geq 0$  then

!  $w = w - x$ ;

! end

! end

computing loss  
i.e., finding  
error made  
and if made we  
will update the  
weights

Make an  
error at negative  
point if the  
is greater than zero  
 $\therefore$  we will update  
the weights

// The Algorithm converges when all  
the weights are classified correctly

Algorithm of Perceptron

Based on the above Algorithm:

Let  $W = [w_1, w_2, w_3, \dots]$   
 $\cdot X = [x_1, x_2, x_3, \dots]$

$$\cos \theta = \frac{W \cdot X}{\|W\| \|X\|} = \frac{\sum w_i x_i}{\|W\| \|X\|}$$

Always positive denominator

$-1 \leq \cos \theta \leq 1$   
 $180^\circ \quad \theta \quad 0$   
 $180^\circ \leftrightarrow 90^\circ \leftrightarrow 0$

The denominator being positive the sign depends on the dot product  $w \cdot x$

For  $x$  belongs to  $p$  if  $w \cdot x < 0$  (Actually we should have  $w \cdot x \geq 0$  we check negative case) then it means that the angle ( $\alpha$ ) between this  $X$  and the current  $W$  greater than  $90$  (but we want to be less than  $90$ ) what new angle happens after update.

Therefore,  $w(\text{new}) = w + x$

$$\begin{aligned}
 \cos(\angle_{\text{new}}) &\propto w_{\text{new}}^T x \\
 &\propto (w + x)^T x \\
 &\propto w^T x + x^T x \\
 &\propto \cos \angle + x^T x \\
 \cos(\angle_{\text{new}}) &\propto \cos \angle + x^T x \quad \cos(\angle_{\text{new}}) > \cos \angle
 \end{aligned}$$

Cosine value increases and the angle between vectors decreases.

Similarly, for the negative for  $x$  belongs to  $N$  if  $w \cdot x \geq 0$  (Actually we need it to be  $w \cdot x < 0$  we see

negative) then it means that angle ( $\alpha$ ) between this  $X$  and the current  $w$  is less than  $0$  (but we want ( $\alpha$ ) to be greater)

$$\begin{aligned}\cos(\angle_{new}) &= \frac{w_{new}^T X}{\|w_{new}\| \|X\|} \\ &= \frac{(w - \alpha X)^T X}{\|w - \alpha X\| \|X\|} \\ &= \frac{w^T X - \alpha X^T X}{\|w - \alpha X\| \|X\|} \\ &= \frac{\cos \angle - \alpha X^T X}{\|w - \alpha X\| \|X\|} \\ \cos(\angle_{new}) &< \cos \angle\end{aligned}$$

Cosine value decreases the angle between vectors increases.

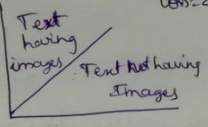
Perceptron learning Algorithm will only work on data which is linearly separable the perceptron learning algorithm will not work if data is not linearly separable.

### Evaluation:

Accuracy formula is the ratio of Number of correct predictions to the total number of predictions.

Accuracy =  $\frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$

An Edge on Capstone Project :- (The simplest model on binary classification)

Data	Task	Model	Loss	Learning	Evaluation
collection of dataset	Boolean Text/ Netext		$\text{loss} = \sum (y_i - \hat{y}_i)$	Perceptron Learning Algorithm	Accuracy = $\frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$

This is the detailed explanation of about the perceptron model.

This is a small try ,uploading the notes . I believe in **"Sharing knowledge is that best way of developing skills"**.Comments will be appreciated. Even small edits can be suggested.

Each Applause will be a great encouragement.

**Do follow my medium for more updates.....**